Эффективное объектно-ориентированное программирования

3-е издание
охватывает Python 2.5

Изучаем

Python

СИМВОЛ®
O'REILLY®

Марк Лутц

Learning Python

Third Edition

Mark Lutz

O'REILLY®

Изучаем Python

Третье издание

Марк Лутц

Санкт-Петербург — Москва
2009

Марк Лутц

Изучаем Python, 3-е издание

Перевод А. Киселева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Научный редактор	*Б. Попов*
Редактор	*Ю. Бочина*
Корректор	*С. Минин*
Верстка	*Д. Орлова*

Лутц М.

Изучаем Python, 3-е издание – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 848 с., ил.

ISBN 978-5-93286-138-7

Мощный, переносимый и легкий в использовании язык программирования Python идеально подходит для разработки самостоятельных программ и сценариев. Эта книга позволит быстро и эффективно овладеть базовыми основами языка Python независимо от степени предварительного знакомства с ним.

Третье издание «Изучаем Python» основано на материалах учебных курсов, которые автор, Марк Лутц, ведет уже на протяжении десяти лет. В книге представлены основные типы объектов в языке Python, порядок их создания и работы с ними, а также функции как основной процедурный элемент языка. Рассматриваются методы работы с модулями и дополнительными объектно-ориентированными инструментами языка Python – классами. Включены описания моделей и инструкций обработки исключений, а также обзор инструментов разработки, используемых при создании крупных программ. Обсуждаются изменения в ожидаемой версии 3.0. В конце глав представлены упражнения и вопросы, которые позволят применить новые знания на практике и проверить, насколько хорошо усвоен материал.

Если вы хотите понять, почему выбирают Python такие компании, как Google и Intel, Cisco и Hewlett-Packard, почему этот язык используют в NASA для научных вычислений, то эта книга станет для вас лучшей отправной точкой.

ISBN 978-5-93286-138-7
ISBN 978-0-596-51398-6 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 21.11.2008. Формат 70×100¹/16. Печать офсетная.
Объем 53 печ. л. Тираж 2000 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Посвящается Вере.
Ты жизнь моя.

Оглавление

Предисловие

Эта книга представляет собой введение в язык программирования Python. Python – это популярный язык программирования, используемый как для разработки самостоятельных программ, так и для создания прикладных сценариев в самых разных областях применения. Это мощный, переносимый, простой в использовании и свободно распространяемый язык программирования.

Цель этой книги – позволить вам быстро овладеть основными принципами базового языка Python независимо от уровня вашей подготовки. Прочитав эту книгу, вы получите объем знаний, достаточный для использования Python в самых разных областях.

О третьем издании

За четыре года, прошедшие с момента выхода второго издания книги в 2003 году, произошли существенные изменения как в самом языке Python, так и в темах, которые я представляю на своих курсах обучения языку Python. Я постарался оставить как можно больше из предыдущего издания, кроме того, это новое издание отражает изменения последнего времени, появившиеся в самом языке Python и в обучении языку Python. Помимо этого была несколько изменена структура книги.

Изменения в языке Python

Если говорить о версии языка, это издание описывает Python 2.5 и отражает все изменения, появившиеся в языке с момента выхода второго издания книги. (Во втором издании описывался язык Python 2.2 и некоторые нововведения версии 2.3.) Кроме того, в этом издании обсуждаются изменения, которые ожидаются в версии Python 3.0. Ниже приводится список основных тем, касающихся языка программирования, которые вы найдете в этом издании:

- Новая условная конструкция `B if A else C` (глава 12).
- Оператор контекста `with/as` (глава 27).
- Унификация конструкции `try/except/finally` (глава 27).
- Синтаксис относительного импорта (глава 21).
- Выражения-генераторы (глава 17).
- Новые особенности функций-генераторов (глава 17).

- Функции-декораторы (глава 26).
- Объектный тип множества (глава 5).
- Новые встроенные функции: `sorted`, `sum`, `any`, `all`, `enumerate` (главы 4 и 13).
- Объектный тип десятичных чисел с фиксированной точностью представления (глава 5).
- Расширенные сведения о файлах, списках, итераторах и многом другом (главы 13 и 17).
- Новые инструменты разработки: Eclipse, `dustutils`, `unittest` и `doctest`, расширения IDLE, Shedskin и т. д. (главы 3 и 29).

Менее значительные изменения в языке (такие, как широко используемые значения `True` и `False`, новая функция `sys.exc_info`, которая возвращает подробную информацию об исключении, и обработка строковых исключений, методы для работы со строками и встроенные функции `apply` и `reduce`) обсуждаются на протяжении всей книги. Кроме того, здесь приводится расширенное описание некоторых новых особенностей, впервые появившихся в предыдущем издании, включая третье измерение при работе со срезами и передачу произвольного числа аргументов функции, включая функцию `apply`.

Изменения в обучении языку Python

Кроме уже перечисленных изменений в самом языке это издание книги дополнено новыми темами и примерами, наработанными мною при преподавании на курсах обучения языку Python в последние годы. Например, здесь вы найдете:

- Новую главу о встроенных типах (глава 4).
- Новую главу о синтаксических конструкциях (глава 10).
- Полностью новую главу о динамической типизации с углубленным освещением этого вопроса (глава 6).
- Расширенное введение в ООП (глава 22).
- Новые примеры работы с файлами, областями видимости, вложенными инструкциями, классами, исключениями и т. д.

Множество изменений и дополнений было сделано, чтобы облегчить чтение книги начинающим программистам. Учитывая опыт преподавания языка Python на курсах, обсуждение некоторых моментов было перенесено в другие, более соответствующие тематике разделы. Например, описание списков и итераторов теперь приводится вместе с описанием оператора цикла `for`, а не с описанием функциональных инструментов, как это было ранее.

Поскольку книга фактически превратилась в стандартный учебник по Python, в этом издании существенно расширено описание основ языка, изложение материала стало более полным и наглядным.

В дополнение в этом издании приводится полностью обновленный комплект советов и рекомендаций, подобранных из опыта преподавания в течение последних 10 лет и практического использования Python в течение последних 15 лет. Также были дополнены и расширены учебные упражнения – с целью отразить наиболее удачные современные приемы программирования на языке Python, его новые особенности и показать наиболее типичные ошибки, которые совершают начинающие программисты на моих курсах. Вообще основы языка в этом издании обсуждаются более широко, чем в предыдущих изданиях, во-первых, потому что сам язык был расширен, а во-вторых, потому что я добавил немало информации, которая имеет важное значение для практического применения.

Структурные изменения в этом издании

Учитывая разросшийся объем книги, материал, как и в предыдущем издании, разбит на несколько частей. Чтобы облегчить усвоение основ языка, весь материал поделен на несколько частей, каждая из которых содержит несколько глав. Например, типы и инструкции теперь описываются в двух разных частях, в каждой из которых основным типам и инструкциям отведены отдельные главы. Новая структура позволяет сказать больше, не перегружая читателя при этом. При переработке материала упражнения и описания наиболее распространенных ошибок были перемещены из конца главы в конец части, и теперь они приводятся в конце последней главы каждой части.

Упражнения, которые приводятся в конце каждой части, я также дополнил в этом издании кратким обзором и контрольными вопросами в конце каждой главы, что должно помочь вам «пролистать» прочитанный материал. Каждая глава заканчивается серией вопросов, которые помогут вам проверить, насколько хорошо вы поняли изложенный материал. В отличие от упражнений в конце каждой части, решения для которых приводятся в приложении В, ответы на вопросы в конце каждой главы следуют непосредственно за вопросами. Я рекомендую просматривать ответы, даже если вы уверены, что правильно ответили на вопросы, потому что эти ответы, кроме всего, являются кратким обзором только что пройденной темы.

Несмотря на наличие новых тем, эта книга по-прежнему ориентирована на тех, кто только начинает знакомство с языком Python. Она задумывалась как учебник по Python для программистов[1]. Во многом она унаследовала из первых двух изданий и материал, и структуру, и акценты подачи материала. Где это необходимо, я расширил вводную

[1] Под «программистами» я подразумеваю всех, кто в прошлом написал хотя бы одну строчку программного кода на любом языке программирования. Если вы не относитесь к этой категории, эта книга все равно будет вам полезна, но вы должны знать, что она в основном обучает языку Python, а не основам программирования.

часть для начинающих и отделил более сложные темы от основного потока обсуждения, чтобы не усложнять описание основ. Поскольку это издание в значительной степени основано на проверенном временем опыте преподавания, оно, как и первые два, может служить вводным курсом для самостоятельного изучения языка Python.

Ограничение области применения книги

Третье издание представляет собой учебник по основам языка программирования Python и ничего больше. Здесь приводятся всесторонние сведения о языке, которые необходимо знать, прежде чем приступать к практическому его использованию. Материал подается в порядке постепенного усложнения и дает полное представление о языке программирования, не фокусируясь на областях его применения.

Для некоторых «изучить Python» означает потратить час-другой на изучение руководств в сети Интернет. Такой подход пригоден для опытных программистов – в конце концов, Python – довольно простой язык по сравнению с другими языками программирования. Однако проблема такого ускоренного изучения состоит в том, что на практике программисты часто сталкиваются с необычными случаями необъяснимого изменения значений переменных, параметров по умолчанию и т. д. Цель этой книги состоит в том, чтобы дать твердое понимание основ языка Python, чтобы даже самые необычные случаи находили свое объяснение.

Это ограничение было введено преднамеренно. Ограничившись обсуждением основ языка, мы можем заняться более глубоким и полным их исследованием. Более полное обсуждение темы прикладного использования Python и справочные материалы, не вошедшие в эту книгу, вы найдете в других публикациях издательства O'Reilly, таких как «Programming Python», «Python Cookbook», «Python in a Nutshell» и «Python Pocket Reference». Цель книги, которую вы сейчас читаете, состоит исключительно в изучении языка Python, чтобы потом вы смогли применять его для решения прикладных задач, независимо от предметной области, в которой вы работаете.

По этой причине некоторые справочные материалы и разделы предыдущих изданий (примерно 15 процентов предыдущего издания) были вырезаны с целью расширить разделы с описанием основ языка Python. Благодаря этому читатели этого издания получат более полное представление об основах языка программирования и – как следствие – более полезный учебник по языку Python. В качестве заключительного упражнения в конце книги (глава 29) приводится несколько усложненных примеров для самостоятельного изучения.

Об этой книге

В этом разделе приводятся некоторые наиболее важные замечания об этой книге в общем, не зависящие от номера издания. Никакая книга

не способна удовлетворить все нужды и потребности читателя, поэтому важно понимать основные цели книги.

Предварительные условия

В действительности книга не предъявляет никаких предварительных условий. Она с успехом использовалась как начинающими программистами, так и умудренными опытом ветеранами. Если у вас есть желание изучать Python, эта книга наверняка поможет вам. Наличие у читателя некоторого опыта в программировании не является обязательным, но будет совсем не лишним.

Эта книга задумывалась как введение в Python для программистов. Возможно, она не идеальна для тех, кто раньше никогда не имел дела с компьютерами (например, мы не будем тратить время, чтобы объяснять, что такое компьютер), но я не делал никаких предположений о наличии у читателя опыта программирования или об уровне его подготовки.

С другой стороны, – я не считаю нужным обижать читателей, предполагая, что они «чайники», что бы это ни означало, – писать полезные программы на языке Python просто, и эта книга покажет, как это делается. В книге Python иногда противопоставляется другим языкам программирования, таким как C, C++, Java™ и Pascal, но эти сравнения можно просто игнорировать, если ранее вам не приходилось работать с этими языками программирования.

Отношения этой книги с другими книгами

Эта книга охватывает все основные аспекты языка программирования Python, но при этом я старался ограничить круг обсуждаемых тем, чтобы уменьшить объем книги. Для сохранения простоты в ней рассматриваются самые основные понятия, используются небольшие и очевидные примеры и опущены некоторые незначительные детали, которые вы найдете в справочных руководствах. По этой причине данная книга скорее должна рассматриваться как введение, как первый шаг к другим, более специализированным и более полным книгам.

Например, мы не будем говорить об интеграции Python/C – это слишком сложная тема, которая, однако, является центральной для многих систем, основанных на применении Python. Мы также не будем говорить об истории развития Python и о процессе его разработки. А таких популярных применений Python, как создание графического интерфейса, разработка системных инструментов и работа с сетью, мы коснемся лишь очень кратко, если они вообще будут упоминаться. Естественно, при таком подходе из поля зрения выпадает значительная часть общей картины.

Вообще говоря, Python стоит на более высоком качественном уровне относительно других языков в мире языков сценариев. Некоторые из его идей требуют более глубокого изучения, чем может вместить эта книга, поэтому с моей стороны было бы ответственно порекомендовать

продолжить его изучение после того, как вы закончите читать эту книгу. Я надеюсь, что большинство читателей продолжит изучение принципов разработки приложений на этом языке, обратившись к другим источникам информации.

Ориентированная в основном на начинающих программистов, книга «Изучаем Python» может быть дополнена другими книгами издательства O'Reilly о языке Python. Например, существует еще одна моя книга «Programming Python», содержащая более объемные и полные примеры наряду с описанием приемов прикладного программирования, которая задумывалась как продолжение книги, которую вы сейчас читаете. Текущие издания книг «Изучаем Python» и «Programming Python» представляют собой две части курса обучения, который преподает автор, – основы языка и прикладное программирование. Кроме того, в качестве справочника можно использовать еще одну книгу издательства O'Reilly, «Pyton Pocket Reference», где приводятся некоторые подробности, опущенные здесь.

Для дальнейшего изучения можно порекомендовать книги, содержащие дополнительные сведения, примеры или особенности использования языка Python в определенных прикладных областях, таких как веб-приложения и создание графических интерфейсов. Например, книги «Python in a Nutshell» (O'Relly) и «Python Essential Reference» (Sams) содержат справочную информацию. Книга «Python Cookbook» (O'Reilly) представляет собой сборник примеров для тех, кто уже знаком с приемами прикладного программирования. Поскольку выбор книг является делом достаточно субъективным, я рекомендую вам самостоятельно поискать такие, которые наиболее полно будут отвечать вашим потребностям. Неважно, какие книги вы выберете, главное чтобы вы помнили, что для дальнейшего изучения Python вам необходимы более реалистичные примеры, чем приводятся здесь.

На мой взгляд, эта книга будет для вас отличным учебником начального уровня, даже несмотря на ее ограниченность (и скорее всего именно поэтому). Здесь вы найдете все, что необходимо знать, прежде чем приступать к созданию программ и сценариев на языке Python. К тому моменту, когда вы закончите чтение этой книги, вы изучите не только сам язык, но и начнете понимать, как лучше применить его к решению ваших повседневных задач. Кроме того, у вас будет все необходимое для изучения более сложных тем и примеров, которые будут встречаться на вашем пути.

Стиль и структура книги

Эта книга основана на материалах трехдневных практических курсов изучения языка Python. В конце каждой главы содержится список контрольных вопросов, а в конце последней главы каждой части – упражнения. Ответы на контрольные вопросы приводятся непосредственно в самих главах, а примеры решения упражнений – в приложении B.

Контрольные вопросы подобраны так, что они представляют собой краткий обзор рассмотренного материала, а упражнения спроектированы так, чтобы сразу же научить вас правильному стилю программирования и, как правило, каждое упражнение соответствует одному из ключевых аспектов курса.

Я настоятельно рекомендую прорабатывать контрольные вопросы и упражнения в ходе чтения книги не только для того, чтобы получить опыт программирования на Python, но и потому, что в упражнениях поднимаются проблемы, которые не обсуждаются нигде в книге. Ответы на вопросы в главах и примеры решения упражнений в приложении В в случае необходимости помогут вам выйти из затруднительных положений (вы можете заглядывать в ответы так часто, как это потребуется).

Общая структура книги также следует структуре учебного курса. Так как эта книга задумывалась как быстрое введение в основы языка программирования, изложение материала организовано так, чтобы оно отражало основные особенности языка, а не частности. Мы будем двигаться от простого к сложному: от встроенных типов объектов к инструкциям, элементам программ и т. д. Каждая глава является полным и самостоятельным описанием одной темы, но каждая последующая глава основана на идеях, введенных в предыдущих главах (например, когда речь пойдет о классах, я буду исходить из предположения, что вы уже знаете, как создаются функции), поэтому для большинства читателей имеет смысл читать книгу последовательно.

В общих чертах, эта книга описывает язык программирования Python при движении от простого к сложному. Каждая часть посвящена отдельной крупной характеристике языка – типам, функциям и т. д. В большинстве своем примеры являются законченными небольшими сценариями (некоторые из них являются достаточно искусственными, но они иллюстрируют достижение поставленной цели). Если быть более точным, здесь вы найдете:

Часть I. *Введение*

Изучение Python мы начнем с общего обзора этого языка и с ответов на очевидно возникающие вопросы: почему кто-то использует этот язык, для решения каких задач он может использоваться и т. д. В первой главе рассматриваются основные идеи, лежащие в основе технологии, которые должны дать вам некоторые начальные представления. Далее начинается сугубо технический материал книги. Здесь мы рассмотрим, как выполняют программы человек и интерпретатор Python. Цель этой части книги состоит в том, чтобы дать вам начальные сведения, которые позволят вам работать с последующими примерами и упражнениями.

Часть II. *Типы и операции*

Далее мы приступим к исследованию языка программирования Python и начнем его изучение с основных типов объектов, таких как числа, списки, словари и т. д. Обладая только этими инструментами,

вы уже сможете писать достаточно сложные программы. Это самая важная часть книги, потому что она закладывает основу для последующих глав. В этой части мы также рассмотрим динамическую типизацию и ссылки – ключевые аспекты языка Python.

Часть III. *Инструкции и синтаксис*

В следующей части вводятся *инструкции* языка Python – программный код на языке Python, который создает и обслуживает объекты. Здесь также будет представлена общая синтаксическая модель Python. Хотя эта часть в основном сосредоточена на описании синтаксиса, тем не менее, здесь приводятся сведения о дополнительных инструментальных средствах, таких как система PyDoc, и рассматриваются альтернативные стили написания программного кода.

Часть IV. *Функции*

В этой части мы начнем рассматривать высокоуровневые способы структурирования программ на языке Python. Функции предоставляют простой способ упаковки программного кода многократного использования и предотвращения появления избыточного кода. В этой части мы исследуем правила видимости программных элементов в языке Python, приемы передачи аргументов и многое другое.

Часть V. *Модули*

Модули Python позволяют организовать наборы инструкций и функций в виде крупных компонентов, и в этой части будет показано, как создавать модули, как их использовать и перезагружать. Здесь мы также рассмотрим некоторые более сложные темы, такие как пакеты модулей, перезагрузка модулей и переменная `__name__`.

Часть VI. *Классы и ООП*

Здесь мы приступим к исследованию объектно-ориентированного программирования (ООП). *Классы* – это необязательный, но очень мощный инструмент структурирования программного кода многократного использования. Здесь вы увидите, что классы по большей части используют идеи, которые будут описаны к этому моменту, а ООП в языке Python в основном представляет собой поиск имен в связанных объектах. Здесь вы также увидите, что объектно-ориентированный стиль программирования в языке Python не является обязательным, но может существенно сократить время разработки, особенно если речь идет о долгосрочных проектах.

Часть VII. *Исключения и инструменты*

Изучение языка мы закончим рассмотрением модели обработки исключительных ситуаций, а также кратким обзором инструментальных средств разработки, которые особенно удобны при разработке крупных программ (например, инструменты отладки и тестирования). Эта часть появляется в последний раз, в следующих версиях все исключения должны быть классами.

Часть VIII. *Приложения*

Книга заканчивается двумя приложениями, где приводятся рекомендации по использованию языка Python на различных платформах (Приложение A) и варианты решения упражнений, которые приводятся в конце каждой части (Приложение B). Ответы на контрольные вопросы, которые приводятся в конце каждой главы, находятся непосредственно в самих главах.

Обратите внимание: предметный указатель и оглавление могут использоваться для поиска информации, но в этой книге нет приложений со справочными материалами (эта книга – учебник, а не справочник). Как уже говорилось выше, в качестве справочников по синтаксису и встроенным особенностям языка Python вы можете использовать книгу «Python Pocket Reference» (O'Reilly) и справочники на сайте *http://www.python.org*.

Обновления книги

Книга продолжает улучшаться (и исправляются оши#%& опечатки). Обновления, дополнения и исправления к этой книге можно найти в сети Интернет на одном из следующих сайтов:

http://www.oreilly.com/catalog/9780596513986/
(веб-страница книги на сайте издательства O'Reilly)

http://www.rmi.net/~lutz (сайт автора книги)

http://www.rmi.net/~lutz/about-lp.html
(веб-страница книги на сайте автора)

Последний из этих трех URL указывает на веб-страницу, где я выкладываю обновления, однако если эта ссылка окажется ошибочной, вам придется воспользоваться поисковой системой, чтобы восстановить ее. Если бы я был ясновидящим, я указал бы точную ссылку, но Интернет меняется быстрее, чем печатаются книги.

О программах в этой книге

Эта книга и все примеры программ в ней основаны на использовании Python 2.5. Я не стремлюсь предсказывать будущее, тем не менее, в ходе изучения мы будем обсуждать некоторые идеи, которые, как ожидается, будут реализованы в версии 3.0.

Однако, так как эта книга описывает базовые основы языка, можно быть уверенным, что большая часть из того, о чем здесь рассказывается, в следующих версиях Python изменится не очень сильно. Большая часть информации из этой книги применима и к более ранним версиям Python, кроме некоторых случаев; и, естественно, в случае использования расширений, которые появятся после выхода этой книги, ничего гарантировать нельзя.

Существует эмпирическое правило: лучшей версией Python является последняя его версия. Так как эта книга описывает основы языка, большинство сведений применимо к Jython – реализации Python на языке Java, а также к другим реализациям, описанным в главе 2.

Исходные тексты примеров, а также ответы к заданиям можно получить на веб-сайте книги по адресу *http://www.oreilly.com/catalog/9780596513986/*. Вас волнует вопрос, как запускать примеры? Он во всех подробностях обсуждается в главе 3, поэтому потерпите до этой главы.

В преддверии выхода Python 3.0

Первая альфа-версия Python 3.0 вышла как раз перед тем, как эта книга была отправлена в печать, уже после того, как она была написана. Официально это издание книги основано на линейке версий Python 2.x (в частности, на версии 2.5), но она была дополнена многочисленными примечаниями об ожидаемых изменениях в версии Python 3.0.

Официальная версия 3.0 выйдет как минимум через год после выхода этой книги и едва ли будет широко использоваться, по меньшей мере, еще два года. Однако, если вы приобрели эту книгу, когда версия 3.0 уже получила широкое распространение, в этом разделе вы найдете краткое описание изменений в языке, которое поможет вам выполнить переход на новую версию.

За редкими исключениями, в основном язык Python 3.0 будет соответствовать описываемому в книге и влияние этих изменений на типичный программный код будет весьма незначительным. То есть основы языка Python, о которых говорится в этой книге, не будут изменяться от версии к версии, и благодаря этому читатели могут с пользой для себя изучать эти основы, прежде чем переходить к рассмотрению особенностей, характерных для конкретных версий.

Однако, чтобы помочь вам в будущем, ниже приводится список основных отличий Python 3.0. Здесь же приводятся ссылки на главы, где обсуждаются или упоминаются эти изменения. Этот список упорядочен по возрастанию номеров глав. Некоторые из этих изменений могут быть реализованы в современной версии Python 2.5, а некоторые нет. Поскольку в данный момент для большинства читателей изучать этот список не имеет большого смысла, я рекомендую сначала прочитать эту книгу, чтобы изучить основы языка Python, а затем вернуться сюда позднее и ознакомиться с грядущими изменениями. В версии Python 3.0:

- Удалена нынешняя встроенная функция `execfile()`. Вместо нее следует использовать функцию `exec()` (глава 3).
- Встроенная функция `reload()` возможно будет удалена. Альтернатива пока не известна (главы 3 и 19).
- Преобразование в строковое представление с помощью обратных кавычек `` `X` `` будет недоступно: используйте функцию `repr(X)` (глава 5).

- Избыточная операция проверки на неравенство `X<>Y` будет удалена: используйте операцию `X!=Y` (глава 5).
- Множества можно будет создавать с использованием синтаксиса литералов `{1, 3, 2}`, что эквивалентно используемой в настоящее время форме записи: `set([1, 3, 2])` (глава 5).
- Множества могут определяться программно: `{f(x) for x in S if P(x)}`, что эквивалентно используемой в настоящее время форме записи выражения-генератора: `set(f(x) for x in S if P(x))` (глава 5).
- Операция деления `X/Y` всегда возвращает число с плавающей точкой, даже если оба операнда являются целыми числами. Чтобы получить нынешнюю операцию деления с усечением дробной части, следует использовать `X//Y` (глава 5).
- Существует единственный целочисленный тип `int`, который обеспечивает точность представления целых чисел, соответствующую нынешнему типу `long` (глава 5).
- Восьмеричные и двоичные литералы: текущая форма записи восьмеричных чисел `0666` будет вызывать ошибку: используйте вместо нее запись в форме `0o666`, соответствующим образом будет изменен и результат, возвращаемый функцией `oct()`. Также запись в форме `0b1010` будет эквивалентна числу 10, а функция `bin(10)` будет возвращать `"0b1010"` (глава 5).
- Строковый тип `str` поддерживает текст Unicode, а для представления строк с однобайтовыми символами создан новый тип `bytes` (например, для случаев, когда текст загружается из файлов в двоичном режиме). Тип `bytes` – это последовательность переменной длины малых целых чисел с интерфейсом, несколько отличающимся от `str` (глава 7).
- Появился новый дополнительный способ форматирования строк, так `"See {0}, {1} and {foo}".format("A", "B", foo="C")` вернет результат `"See A, B and C"` (глава 7).
- Метод словаря `D.has_key(X)` будет удален. Вместо него следует использовать проверку на членство `X in D` (главы 4 и 8).
- Сравнение (при сортировке) смешанных нечисловых типов вместо использования текущей реализации механизма упорядочения будет вызывать исключение (главы 8 и 9).
- Методы словаря `.keys()`, `.items()` и `.values()` вместо списков будут возвращать «представления» объектов, поддерживающие возможность выполнения итераций. Чтобы вернуться к прежней логике выполнения, следует выполнять принудительное преобразование с помощью `list()` (глава 8).
- Согласно предыдущему пункту следующий шаблон программирования будет неработоспособен: `k = D.keys(); k.sort()`; вместо него следует использовать `k = sorted(D)` (главы 4 и 8).
- Встроенная функция `file()` может быть удалена. Вместо нее следует использовать функцию `open()` (глава 9).

- Встроенная функция `raw_input()` будет переименована в `input()`. Чтобы обеспечить поддержку логики работы нынешней функции `input()`, следует использовать `eval(input())` (глава 10).
- Инструкция выполнения строки программного кода `exec` снова станет встроенной функцией (глава 10).
- Появятся новые зарезервированные слова `as`, `with` и `nonlocal`. Согласно предыдущему пункту `exec` перестанет быть зарезервированным словом (глава 11).
- Инструкция печати станет функцией, поддерживающей большое число возможностей. Вместо `print x, y` следует использовать `print(x, y)`, а также можно использовать новые ключевые аргументы функции для настройки параметров печати: `file=sys.stdout`, `sep=" "` и `end="\n"` (глава 11).
- Появится расширенная реализация операции распаковки: инструкция, поддерживающая универсальную форму присваивания последовательности, такая как `a, b, *rest = some_sequence`, теперь будет работать, как и `*rest, a = stuff`. Таким образом, число элементов слева и справа от инструкции присваивания больше не должно совпадать (глава 11).
- Автоматический режим распаковки кортежей через присваивание последовательности для функций будет недоступен. Больше нельзя будет записать `def foo(a, (b, c)):`, вместо этого необходимо будет выполнять явное присваивание последовательности: `def foo(a, bc): b, c = bc` (глава 11).
- Встроенная функция `xrange()` будет переименована в `range()`. То есть в этой версии Python будет существовать только функция `range()` (глава 13).
- В протоколе итераций метод `X.next()` будет переименован в `X.__next__()`, и появится новая встроенная функция `next(X)`, которая будет вызывать метод `X.__next__()` объекта (главы 13 и 17).
- Встроенные функции `zip()`, `map()` и `filter()` будут возвращать итераторы. Чтобы вернуться к логике использования списков, следует применять `list()` (главы 13 и 17).
- Функции могут включать необязательные комментарии, описывающие аргументы и результаты: так, в результате объявления `def foo(x: "spam", y: list(range(3))) -> 42*2:` к объекту-функции на этапе времени выполнения будет присоединен атрибут-словарь `foo.func_annotations: {'x': "spam", 'y': [0, 1, 2], "return": 84}` (глава 15).
- Новая инструкция `nonlocal x, y` позволит выполнять присваивание переменным в области видимости функции (глава 16).
- Функция `apply(func, args, kws)` будет удалена. Вместо нее следует использовать синтаксическую конструкцию `func(*args, **kws)` (главы 16 и 17).

- Встроенная функция `reduce()` будет удалена. Вместо нее организуйте циклы, как показано в этой книге; `lambda`, `map()` и `filter()` будут сохранены в версии 3.0 (глава 17).
- Все импортирование по умолчанию будет выполняться по абсолютному пути, а собственный каталог пакета будет пропускаться: для выполнения импорта по относительному пути, как это делается сейчас, следует использовать синтаксическую конструкцию `from . import` (глава 21).
- Все классы будут классами нового стиля и будут поддерживать современные новые расширения (глава 26).
- Наследование `class Spam(object)`, необходимое для создания нынешних классов, будет необязательным для классов. В версии 3.0 и нынешняя «классика», и классы «нового стиля» будут относиться к тому, что сейчас называется классами нового стиля (глава 26).
- В инструкции `try` конструкция `except name` превратится в конструкцию `except name as value` (глава 27).
- В инструкции `raise` конструкция `raise E, V` должна будет записываться как `raise E(V)`, то есть экземпляр исключения должен создаваться явно (глава 27).
- Включен оператор контекста исключений `with/as`, описываемый в этой книге (глава 27).
- Все пользовательские и встроенные исключения описываются классами, а не строками (глава 28).
- Все исключения, определяемые пользователем, должны наследовать встроенный `BaseEsception` – корневой класс иерархии классов исключений (`Exception` – это его подкласс и его вполне можно использовать в качестве базового для организации своей иерархии). Встроенный класс `StandardException` будет ликвидирован (глава 28).
- Структура пакета стандартной библиотеки может быть существенно изменена (подробности – в примечаниях к выпуску Python 3.0).

Список нововведений может показаться устрашающим на первый взгляд, однако не следует забывать, что основы языка Python, описываемые в этой книге, останутся неизменными и в версии 3.0. В действительности большая часть из того, что перечислено выше, будет оказывать на программистов не слишком большое влияние, если это влияние вообще будет ощущаться.

Кроме того, следует отметить, что этот список по-прежнему остается достаточно гипотетическим и в конечном счете может оказаться неполным и неточным, поэтому за официальной информацией следует ознакомиться с примечаниями к выпуску Python 3.0. Если вы пишете программный код для линейки Python 2.0, обратите внимание на сценарий «2to3», который выполняет автоматическое преобразование программного кода, написанного для интерпретатора версии 2.x в программный

код для интерпретатора версии 3.0. Этот сценарий будет поставляться в составе Python 3.0.

Об этой серии

Книги издательства O'Reilly из серии «Изучаем...» предназначены для тех, кто хочет приобрести новые знания и предпочитает структурный подход к изучению. Каждая книга из этой серии использует принципы обучения, которые мы (не без вашей помощи) выработали для передачи знаний, необходимых вам, чтобы присоединиться к новому проекту, справиться с неожиданным заданием или быстро изучить новый язык программирования.

Чтобы получить максимум пользы от любой книги из серии «Изучаем...», мы рекомендуем прорабатывать главы последовательно. Вы увидите, что усвоение материала идет быстрее, если внимательно читать указания и рекомендации, включенные в них. Вы можете также использовать заключительные разделы для предварительного знакомства с ключевыми аспектами каждой главы и с тем, что вам предстоит изучать. Наконец, чтобы помочь вам проверить степень усвоения материала, каждая глава завершается разделом «Закрепление пройденного», который включает короткие контрольные вопросы. Кроме того, каждая часть включает практические упражнения.

Книги серии «Изучаем...» работают с вами как надежный и проверенный коллега или преподаватель; мы стремимся сделать ваше обучение как можно более приятным. Свои отзывы о нашей работе и предложения по улучшению можете направлять по адресу *learning@oreilly.com*.

Использование программного кода примеров

Данная книга призвана оказать вам помощь в решении ваших задач. Вы можете свободно использовать примеры программного кода из этой книги в своих приложениях и в документации. Вам не нужно обращаться в издательство за разрешением, если вы не собираетесь воспроизводить существенные части программного кода. Например, если вы разрабатываете программу и используете в ней несколько отрывков программного кода из книги, вам не нужно обращаться за разрешением. Однако в случае продажи или распространения компакт-дисков с примерами из этой книги вам *необходимо* получить разрешение от издательства O'Reilly. Если вы отвечаете на вопросы, цитируя данную книгу или примеры из нее, получение разрешения не требуется. Но при включении существенных объемов программного кода примеров из этой книги в вашу документацию вам *необходимо* будет получить разрешение издательства.

Мы приветствуем, но не требуем добавлять ссылку на первоисточник при цитировании. Под ссылкой на первоисточник мы подразумеваем указание авторов, издательства и ISBN. Например: «Learning Python, by Mark Lutz. Copyright 2008 O'Reilly Media, Inc., 978-0-596-51398-6».

За получением разрешения на использование значительных объемов программного кода примеров из этой книги обращайтесь по адресу *permissions@oreilly.com*.

Типографские соглашения

В этой книге приняты следующие соглашения:

Курсив

Курсив применяется для выделения адресов электронной почты, URL, имен файлов и каталогов, а также терминов, когда они упоминаются впервые.

`Моноширинный шрифт`

Применяется для представления содержимого файлов, вывода команд, а также для выделения имен модулей, методов, инструкций и команд.

`Моноширинный жирный`

Используется для выделения команд или текста, который должен быть введен пользователем, а также для выделения участков программного кода в листингах.

`Моноширинный курсив`

Обозначает замещаемые элементы в программном коде и комментарии.

`<Моноширинный шрифт>`

Таким способом выделяются синтаксические элементы, которые должны замещаться действительным программным кодом.

Так выделяются советы, предложения или примечания общего характера, имеющие отношение к расположенному рядом тексту.

Так выделяются предупреждения или предостережения, имеющие отношение к расположенному рядом тексту.

В примерах этой книги символ `%` в начале системной командной строки обозначает приглашение к вводу независимо от того, какое приглашение используется на вашей машине (например, `C:\Python25>` в окне Dos). Вам не нужно вводить символ `%`. Точно также в листингах, отображающих сеанс работы с интерпретатором, не нужно вводить символы `>>>` и `...`, которые показаны в начале строки, – это приглашения к вводу, которые выводятся интерпретатором Python. Вводите лишь текст, который находится сразу же за этими приглашениями. Чтобы помочь вам запомнить это правило, все, что должно вводиться пользователем, выделено жирным шрифтом. Кроме того, обычно не требуется вводить текст в листингах, начинающийся с символа `#`, так как это комментарии, а не исполняемый программный код.

Safari® Books Online

Если на обложке технической книги есть пиктограмма «Safari® Books Online», это означает, что книга доступна в Сети через O'Reilly Network Safari Bookshelf.

Safari предлагает намного лучшее решение, чем электронные книги. Это виртуальная библиотека, позволяющая без труда находить тысячи лучших технических книг, вырезать и вставлять примеры кода, загружать главы и находить быстрые ответы, когда требуется наиболее верная и свежая информация. Она свободно доступна по адресу *http://safari.oreilly.com*.

Как с нами связаться

С вопросами и предложениями, касающимися этой книги, обращайтесь в издательство:

O'Reilly Media
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (в Соединенных Штатах Америки или в Канаде)
707-829-0515 (международный)
707-829-0104 (факс)

Список опечаток, файлы с примерами и другую дополнительную информацию вы найдете на сайте книги:

http://www.oreilly.com/catalog/9780596513986

Свои пожелания и вопросы технического характера отправляйте по адресу:

bookquestions@oreilly.com

Дополнительную информацию о книгах, обсуждения, Центр ресурсов издательства O'Reilly вы найдете на сайте:

http://www.oreilly.com

Обновления и дополнения к книге вы также можете найти на сайтах, упоминавшихся выше в этом предисловии.

Благодарности

Учитывая, что я работаю уже над третьим изданием этой книги, я не могу не пребывать в настроении, что «сложное задание выполнено». Я использовал и пропагандировал Python на протяжении 15 лет и обучал этому языку 10 лет. Несмотря на то, что все течет и все меняется, я по-прежнему поражаюсь успеху, который сопутствует языку Python. В 1992 году большинство из нас едва ли могло предполагать, какими путями он будет развиваться. Но, чтобы не выглядеть безнадежно эго-

центричным, я хотел бы немного вспомнить о прошлом и сказать несколько слов благодарности.

Это была длинная и извилистая дорога. Оглядываясь назад, когда в 1992 году я открыл для себя Python, я предположить не мог, какое влияние он будет оказывать на мою жизнь в следующие 15 лет. Через два года после начала работы над первым изданием «Learning Python» в 1995 году я начал путешествовать по стране и миру, обучая начинающих программистов этому языку программирования. После выхода первого издания в 1999 году преподавание Python и работа над книгами стали моей основной работой, во многом благодаря интенсивному росту популярности Python.

В середине 2007 года, когда я пишу эти слова, мною уже написано девять книг о Python; опыт преподавания Python насчитывает более 10 лет. Проведя более 200 курсов в США, Европе, Канаде и Мексике, я обучил более 3000 студентов. Помимо множества мучительных часов, проведенных в самолетах, эти курсы дали мне возможность существенно улучшить эту и другие книги о Python. За эти годы преподавание помогало улучшать качество книг, а работа над книгами – качество преподавания. Фактически книга, которую вы читаете, была почти полностью получена из программы моих курсов.

В связи с этим я хотел бы поблагодарить всех студентов, которые участвовали в моих курсах на протяжении последних 10 лет. Как развитие языка Python, так и ваши отзывы сыграли немаловажную роль в становлении этой книги. (Нет ничего более поучительного, чем наблюдение за 3000 студентами, которые совершают одни и те же ошибки, свойственные начинающим программистам!) Это издание стало возможным в первую очередь благодаря тем, кто посещал мои курсы после 2003 года, и тем не менее, все, кто посещал мои курсы, начиная с 1997 года, так или иначе помогли в улучшении этой книги. Я особенно хотел бы поблагодарить компании, предоставившие помещения для проведения курсов в Дублине, Мехико, Барселоне, Лондоне, Эдмонтоне и Пуэрто Рико; лучшие условия аренды трудно себе представить.

Я также хотел бы выразить свою благодарность всем, кто принимал участие в производстве этой книги. Редакторам, работавшим над этим проектом: Татьяне Апанди (Tatiana Apandi), работавшей над этим изданием, и многим другим редакторам, работавшим над предыдущими изданиями. Лизе Дейли (Liza Daly) за участие в техническом обзоре этой книги. И издательству O'Reilly, за то что я получил шанс работать над этими девятью проектами книг – это было здорово (правда, порой я чувствовал себя как герой фильма «День сурка»).

Я хочу поблагодарить своего первого соавтора Дэвида Ашера (David Ascher) за его работу над ранними изданиями этой книги. Дэвид написал часть «Outer Layers (Внешние уровни)» для предыдущих изданий, которую мы, к сожалению, убрали из книги, чтобы освободить место для новых материалов об основах языка Python. В это издание

я добавил больше усложненных программ и заключительное упражнение для самостоятельного изучения, но это не компенсирует все то, что пришлось вырезать. Если вам не хватает этого материала, прочитайте ранее приведенные в предисловии примечания по поводу книг, описывающих вопросы прикладного программирования.

За создание такого замечательного и полезного языка я должен поблагодарить Гвидо ван Россума (Guido van Rossum) и все сообщество разработчиков и пользователей Python. Подобно большинству программных продуктов с открытыми исходными текстами Python развивается благодаря героическим усилиям многих программистов. Обладая 15-летним опытом программирования на Python, я по-прежнему нахожу его серьезной забавой. Мне очень повезло, что я смог наблюдать, как из маленького младенца в семействе языков сценариев Python вырос в популярный инструмент, которым пользуются практически все компании, занимающиеся разработкой программного обеспечения. Участвовать в этом процессе было очень волнующим занятием, и поэтому я хотел бы поблагодарить и поздравить с достигнутыми успехами все сообщество Python.

Я также хотел бы поблагодарить своего первого редактора из издательства O'Reilly, Фрэнка Уиллисона (Frank Willison). Идея этой книги во многом принадлежит Фрэнку, и в ней нашли отражение его взгляды. Оглядываясь назад, можно заметить, что Фрэнк оказал существенное влияние как на мою карьеру, так и на Python. Не будет преувеличением сказать, что успех развития Python на начальном этапе в определенной степени обусловлен влиянием Фрэнка. Мы по-прежнему скучаем по нему.

В заключение хотелось бы выразить личную благодарность. Компании OQO за самые лучшие игрушки, какие я только видел. Покойному Карлу Сагану (Carl Sagan), за то что вдохновил 18-летнего мальчишку из Висконсина. Йону Стюарту (Jon Stewart) и Майклу Муру (Michael Moore) за патриотизм. И всем крупным корпорациям, с которыми мне приходилось иметь дело, за то, что напоминают мне о том, как это здорово – работать на самого себя.

Моим детям, Майку (Mike), Сэмми (Sammy) и Рокси (Roxy), за любую будущность, которую они выберут. Вы были детьми, когда я начинал работать с языком Python, и вы каким-то образом выросли за это время; я горжусь вами. Жизнь может перекрыть нам все пути, но путь домой всегда остается открытым.

Но больше всего я благодарен Вере (Vera), моему лучшему другу, моей подруге и моей жене. День, когда я нашел тебя, был лучшим днем в моей жизни. Я не знаю, что принесут мне следующие 50 лет, но я знаю, что хотел бы прожить их рядом с тобой.

Марк Лутц (Mark Lutz)
Бертуд, Колорадо
Июль 2007

I

Введение

1

Python в вопросах и ответах

Если вы купили эту книгу, вы, скорее всего, уже знаете, что такое Python и насколько важно овладеть этим инструментом. Если это не так, вы наверняка не сможете зарабатывать на Python, пока не изучите язык, прочитав оставшуюся часть этой книги, и не напишете пару проектов. Но прежде чем мы приступим к изучению деталей, давайте сначала рассмотрим основные причины высокой популярности языка Python. Перед тем как приступить собственно к языку Python, в этой главе рассматриваются некоторые вопросы, которые обычно задают начинающие программисты, и даются ответы на них.

Почему программисты используют Python?

Это самый типичный вопрос, который задают начинающие программисты, потому что на сегодняшний день существует масса других языков программирования. Учитывая, что число пользователей Python составляет порядка 1 миллиона человек, достаточно сложно однозначно ответить на этот вопрос. Выбор средств разработки иногда зависит от уникальных особенностей и личных предпочтений.

Однако после обучения примерно 200 групп и 3000 студентов за последние 10 лет у меня накопились некоторые общие мысли по этому поводу. Основные факторы, которые приводятся пользователями Python, примерно таковы:

Качество программного обеспечения

Для многих основное преимущество языка Python заключается в удобочитаемости, ясности и более высоком качестве, отличающими его от других инструментов в мире языков сценариев. Программный код на языке Python читается легче, а значит, многократное его использование и обслуживание выполняется гораздо проще, чем использование программного кода на других языках сценариев.

Единообразие оформления программного кода на языке Python облегчает его понимание даже для тех, кто не участвовал в создании этого кода. Кроме того, Python поддерживает самые современные механизмы многократного использования программного кода, каким является объектно-ориентированное программирование (ООП).

Высокая скорость разработки

По сравнению с компилирующими или строго типизированными языками, такими как C, C++ и Java, Python во много раз повышает производительность труда разработчика. Объем программного кода на языке Python обычно составляет треть или даже пятую часть эквивалентного программного кода на языке C++ или Java. Это означает меньший объем ввода с клавиатуры, меньшее количество времени на отладку и меньший объем трудозатрат на сопровождение. Кроме того, программы на языке Python запускаются сразу же, минуя длительные этапы компиляции и связывания, необходимые в некоторых других языках программирования, что еще больше увеличивает производительность труда программиста.

Переносимость программ

Большая часть программ на языке Python выполняется без изменений на всех основных платформах. Перенос программного кода из операционной системы Linux в Windows обычно заключается в простом копировании программного кода сценария с одной машины на другую. Более того, Python предоставляет массу возможностей по созданию переносимых графических интерфейсов, программ доступа к базам данных, веб-приложений и многих других типов программ. Даже интерфейсы операционных систем, включая способ запуска программ и обработку каталогов, в языке Python реализованы переносимым способом.

Поддержка библиотек

В составе Python поставляется большое число собранных и переносимых функциональных возможностей, известных как *стандартная библиотека*. Эта библиотека предоставляет массу возможностей, востребованных в прикладных программах, начиная от поиска текста по шаблону и заканчивая сетевыми функциями. Кроме того, Python допускает расширение, как за счет ваших собственных библиотек, так и за счет библиотек, созданных сторонними разработчиками. Из числа сторонних разработок можно назвать инструменты создания веб-сайтов, программирование математических вычислений, доступ к последовательному порту, разработку игровых программ и многое другое. Например, расширение NumPy позиционируется как свободный и более мощный эквивалент системы программирования математических вычислений Mathlab.

Интеграция компонентов

Сценарии Python легко могут взаимодействовать с другими частями приложения благодаря различным механизмам интеграции.

Эта интеграция позволяет использовать Python для настройки и расширения функциональных возможностей программных продуктов. На сегодняшний день программный код на языке Python имеет возможность вызывать функции из библиотек на языке C/C++, сам вызываться из программ, написанных на языке C/C++, интегрироваться с Java-компонентами, взаимодействовать с такими платформами, как COM и .NET, и производить обмен данными по сети с помощью таких протоколов, как SOAP, XML-RPC и CORBA. Python не является обособленным инструментом.

Удовольствие

Благодаря непринужденности языка Python и наличию встроенных инструментальных средств процесс программирования может приносить удовольствие. На первый взгляд это трудно назвать преимуществом, тем не менее, удовольствие, получаемое от работы, напрямую влияет на производительность труда.

Из всех перечисленных факторов наиболее существенными для пользователей являются первые два (качество и производительность).

Качество программного обеспечения

По своей природе Python имеет простой, удобочитаемый синтаксис и ясную модель программирования. Согласно лозунгу, выдвинутому на недавней конференции по языку Python, основное его преимущество состоит в том, что Python «каждому по плечу» – характеристики языка взаимодействуют ограниченным числом непротиворечивых способов и естественно вытекают из небольшого круга базовых концепций. Это делает язык простым в освоении, понимании и запоминании. Программистам, использующим язык Python, почти не приходится прибегать к справочным руководствам – это непротиворечивая система, на выходе которой получается профессиональный программный код.

Философия Python диктует использование минималистского подхода. Это означает, что даже при наличии нескольких вариантов решения задачи в этом языке обычно существует всего один очевидный путь, небольшое число менее очевидных альтернатив и несколько взаимосвязанных вариантов организации взаимодействий. Более того, Python не принимает решения за вас, когда порядок взаимодействий неочевиден – предпочтение отдается явному описанию, а не «волшебству». В терминах Python явное лучше неявного, а простое лучше сложного.[1]

---

[1] Чтобы получить более полное представление о философии Python, введите в командной строке интерпретатора команду `import this` (как это сделать, будет рассказано в главе 2). В результате будет активизировано «пасхальное яйцо», скрытое в недрах Python, – сборник принципов проектирования, лежащих в основе Python. Аббревиатура EIBTI, происходящая от фразы «explicit is better than implicit» («явное лучше неявного»), превратилась в модное жаргонное словечко.

Помимо философии Python обладает такими возможностями, как модульное и объектно-ориентированное программирование, что естественно упрощает возможность многократного использования программного кода. Поскольку качество находится в центре внимания самого языка Python, оно также находится в центре внимания программистов.

Высокая скорость разработки

Во время бума развития Интернета во второй половине 1990-х годов, было сложно найти достаточное число программистов для реализации программных проектов – от разработчиков требовалось писать программы со скоростью развития Интернета. Теперь, в эпоху экономического спада, картина изменилась. Сегодня от программистов требуется умение решать те же задачи меньшим числом сотрудников.

В обоих этих случаях Python блистал как инструмент, позволяющий программистам получать большую отдачу при меньших усилиях. Он изначально оптимизирован для достижения *высокой скорости разработки* – простой синтаксис, динамическая типизация, отсутствие этапа компиляции и встроенные инструментальные средства позволяют программистам создавать программы за меньшее время, чем при использовании некоторых других инструментов. В результате Python увеличивает производительность труда разработчика во много раз по сравнению с традиционными языками программирования. Это значительное преимущество, которое с успехом может использоваться как во время бума, так и во время спада, а также во время любого промежуточного этапа развития индустрии программного обеспечения.

Является ли Python «языком сценариев»?

Python – это многоцелевой язык программирования, который зачастую используется для создания сценариев. Обычно он определяется как *объектно-ориентированный язык сценариев* – такое определение смешивает поддержку ООП с общей ориентацией на сценарии. Действительно, для обозначения файлов с программным кодом на языке Python программисты часто используют слово «сценарий» вместо слова «программа». В этой книге термины «сценарий» и «программа» рассматриваются как взаимозаменяемые, с некоторым предпочтением термина «сценарий» для обозначения простейших программ, помещающихся в единственный файл, и термина «программа» для обозначения более сложных приложений, программный код которых размещается в нескольких файлах.

Термин «язык сценариев» имеет множество различных толкований. Некоторые предпочитают вообще не применять его к языку Python. У большинства термин «язык сценариев» вызывает три разных ассоциации, из которых одни более применимы к языку Python, чем другие:

Командные оболочки

Иногда, когда кто-то слышит, что Python – это язык сценариев, то представляет себе Python как инструмент для создания системных сценариев. Такие программы часто запускаются из командной строки с консоли и решают такие задачи, как обработка текстовых файлов и запуск других программ.

Программы на языке Python способны решать такие задачи, но это лишь одна из десятков прикладных областей, где может применяться Python. Это не только язык сценариев командной оболочки.

Управляющий язык

Другие пользователи под названием «язык сценариев» понимают «связующий» слой, который используется для управления (то есть для описания сценария работы) другими прикладными компонентами. Программы на языке Python действительно нередко используются в составе крупных приложений. Например, при проверке аппаратных устройств программы на языке Python могут вызывать компоненты, осуществляющие низкоуровневый доступ к устройствам. Точно так же программы могут запускать программный код на языке Python для поддержки настройки программного продукта у конечного пользователя, что ликвидирует необходимость поставлять и пересобирать полный объем исходных текстов.

Простота языка Python делает его весьма гибким инструментом управления. Тем не менее, технически – это лишь одна из многих ролей, которые может играть Python. Многие программисты пишут на языке Python автономные сценарии, которые не используют какие-либо интегрированные компоненты. Это не только язык управления.

Удобство в использовании

Пожалуй, лучше всего представлять себе термин «язык сценариев» как обозначение простого языка, используемого для быстрого решения задач. Это особенно верно, когда термин применяется к языку Python, который позволяет вести разработку гораздо быстрее, чем компилирующие языки программирования, такие как C++. Ускоренный цикл разработки способствует применению зондирующего, поэтапного стиля программирования, который следует попробовать, чтобы оценить по достоинству.

Не надо заблуждаться, Python предназначен не только для решения простых задач. Скорее он упрощает решение задач, благодаря своей простоте и гибкости. Язык Python имеет небольшой набор возможностей, но он позволяет создавать программы неограниченной сложности. По этой причине Python обычно используется как для быстрого решения тактических, так и для решения долговременных, стратегических задач.

Итак, является ли Python языком сценариев? Ответ зависит от того, к кому обращен вопрос. Вообще термин «создание сценариев», вероят-

но, лучше использовать для описания быстрого и гибкого способа разработки, который поддерживается языком Python, а не для описания прикладной области программирования.

Все хорошо, но есть ли у него недостатки?

После 15 лет работы с языком Python и 10 лет преподавания единственный недостаток, который мне удалось обнаружить, – это скорость выполнения программ, которая не всегда может быть такой же высокой, как у программ, написанных на компилирующих языках программирования, таких как C или C++.

Подробнее о концепциях реализации мы поговорим ниже в этой книге. В двух словах замечу, что в современной реализации Python компилирует (то есть транслирует) инструкции исходного программного кода в промежуточное представление, известное как *байт-код*, и затем интерпретирует этот байт-код. Байт-код обеспечивает переносимость программ, поскольку это платформонезависимый формат. Однако, из-за того что Python не создает двоичный машинный код (например, машинные инструкции для микропроцессора Intel), некоторые программы на языке Python могут работать медленнее своих аналогов, написанных на компилирующих языках, таких как C.

Будет ли вас когда-нибудь *волновать* разница в скорости выполнения программ, зависит от того, какого рода программы вы пишете. Python многократно подвергался оптимизации, и в отдельных прикладных областях программный код на этом языке отличается достаточно высокой скоростью выполнения. Кроме того, когда в сценарии Python делается что-нибудь «значительное», например обрабатывается файл или конструируется графический интерфейс, ваша программа фактически выполняется со скоростью, которую способен дать язык C, потому что такого рода задачи решаются компилированным с языка C программным кодом, лежащим в недрах интерпретатора Python. Гораздо важнее, что преимущество в скорости разработки порой важнее потери скорости выполнения, особенно если учесть быстродействие современных компьютеров.

Тем не менее, даже при высоком быстродействии современных процессоров остаются такие области, где требуется максимальная скорость выполнения. Реализация математических вычислений и анимационных эффектов, например, часто требуют наличия базовых вычислительных компонентов, которые решают свои задачи со скоростью языка C (или еще быстрее). Если вы работаете как раз в такой области, вы все равно сможете использовать Python, достаточно лишь выделить из приложения компоненты, требующие максимальной скорости работы, в виде *компилированных расширений* и связать их системой сценариев на языке Python.

В этой книге мы не будем обсуждать расширения слишком подробно, но это один из примеров, когда Python может играть упоминавшуюся

выше роль языка управления. Типичным примером такой двуязычной стратегии может служить расширение *NumPy*, содержащее реализацию математических вычислений для Python; благодаря комбинированию компилированных и оптимизированных библиотек расширения с языком Python, NumPy превращает Python в мощный, эффективный и удобный инструмент математических вычислений. Возможно, вам никогда не придется создавать подобные расширения, но вы должны знать, что в случае необходимости они могут предоставить в ваше распоряжение мощный механизм оптимизации.

Кто в наше время использует Python?

К моменту, когда я пишу эти строки (2007 год), наиболее правдоподобной оценкой числа пользователей Python является число, близкое к 1 миллиону человек во всем мире (с небольшой погрешностью). Эта оценка основана на различных статистических показателях, таких как количество загрузок и результаты опросов разработчиков. Дать более точную оценку достаточно сложно, потому что Python является открытым программным обеспечением – для его использования не требуется проходить лицензирование. Более того, Python по умолчанию включается в состав дистрибутивов Linux, поставляется вместе с компьютерами Macintosh и некоторыми другими программными и аппаратными продуктами, что существенно искажает оценку числа пользователей.

Вообще же количество пользователей Python значительно больше и вокруг него сплотилось очень активное сообщество разработчиков. Благодаря тому что Python появился более 15 лет тому назад и получил широкое распространение, он отличается высокой стабильностью и надежностью. Python используется не только отдельными пользователями, он также применяется настоящими компаниями для создания продуктов, приносящих настоящую прибыль. Например:

- Компания Google широко использует Python в своей поисковой системе и оплачивает труд создателя Python.
- Служба коллективного использования видеоматериалов YouTube в значительной степени реализована на языке Python.
- Популярная программа BitTorrent для обмена файлами в пиринговых сетях (peer-to-peer) написана на языке Python.
- Такие компании, как Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm и IBM используют Python для тестирования аппаратного обеспечения.
- Такие компании, как Industrial Light & Magic, Pixar и другие используют Python в производстве анимационных фильмов.
- Компании JPMorgan Chase, UBS, Getco и Citadel применяют Python для прогнозирования финансового рынка.
- NASA, Los Alamos, Fermilab, JPL и другие используют Python для научных вычислений.

- iRobot использует Python в разработке коммерческих роботизированных пылесосов.
- ESRI использует Python в качестве инструмента настройки своих популярных геоинформационных программных продуктов под нужды конечного пользователя.
- NSA использует Python для шифрования и анализа разведданных.
- В реализации почтового сервера IronProt используется более 1 миллиона строк программного кода на языке Python.
- Проект «ноутбук каждому ребенку» (One Laptop Per Child, OLPC) строит свой пользовательский интерфейс и модель функционирования на языке Python.

И так далее. Пожалуй, единственное, что объединяет все эти компании, – это то, что для решения широкого спектра задач прикладного программирования используется язык программирования Python. Универсальная природа языка обеспечивает возможность его применения в самых разных областях. Фактически, с определенной долей уверенности можно утверждать, что Python так или иначе используется практически каждой достаточно крупной организацией, занимающейся разработкой программного обеспечения, – как для решения краткосрочных тактических задач, так и для разработки долгосрочных стратегических проектов. Как оказалось, Python прекрасно зарекомендовал себя в обоих случаях.

За дополнительными сведениями о компаниях, использующих Python, обращайтесь на веб-сайт *http://www.python.org*.

Что можно делать с помощью Python?

Кроме того, будучи удачно спроектированным языком программирования, Python прекрасно подходит для решения реальных задач из разряда тех, которые разработчикам приходится решать ежедневно. Он используется в самом широком спектре применений – и как инструмент управления другими программными компонентами, и для реализации самостоятельных программ. Фактически, круг ролей, которые может играть Python как многоцелевой язык программирования, практически не ограничен: он может использоваться для реализации всего, что угодно, – от веб-сайтов и игровых программ до управления роботами и космическими кораблями.

Однако сферу использования Python в настоящее время можно разбить на несколько широких категорий. Следующие несколько разделов описывают наиболее типичные области применения Python в наши дни, а также инструментальные средства, используемые в каждой из областей. У нас не будет возможности заняться исследованием инструментов, упоминаемых здесь. Если какие-то из них заинтересуют вас, обращайтесь на веб-сайт проекта Python за более подробной информацией.

Системное программирование

Встроенные в Python интерфейсы доступа к службам операционных систем делают его идеальным инструментом для создания переносимых программ и утилит системного администрирования (иногда они называются *инструментами командной оболочки*). Программы на языке Python могут отыскивать файлы и каталоги, запускать другие программы, производить параллельные вычисления с использованием нескольких процессов и потоков и делать многое другое.

Стандартная библиотека Python обеспечивает возможность связывания в соответствии с требованиями стандартов POSIX и поддерживает все типичные инструменты операционных систем: переменные окружения, файлы, сокеты, каналы, процессы, многопоточную модель исполнения, поиск по шаблону с использованием регулярных выражений, аргументы командной строки, стандартные интерфейсы доступа к потокам данных, запуск команд оболочки, дополнение имен файлов и многое другое. Кроме того, системные интерфейсы в языке Python созданы переносимыми, например, сценарий копирования дерева каталогов не требует внесения изменений, в какой бы операционной системе он ни использовался.

Графический интерфейс

Простота Python и высокая скорость разработки делают его отличным средством разработки графического интерфейса. В состав Python входит стандартный объектно-ориентированный интерфейс к Tk GUI API, который называется *Tkinter,* позволяющий программам на языке Python реализовать переносимый графический интерфейс с внешним видом, присущим операционной системе. Графические интерфейсы на базе Python/Tkinter без изменений могут использоваться в MS Windows, X Window (в операционных системах UNIX и Linux) и Mac OS (как в классической версии, так и в OS X). Свободно распространяемый пакет расширения *PMW* содержит дополнительные визуальные компоненты для набора Tkinter. Кроме того, существует прикладной интерфейс *wxPython GUI API*, основанный на библиотеке C++, который предлагает альтернативный набор инструментальных средств построения переносимых графических интерфейсов на языке Python.

Инструменты высокого уровня, такие как *PythonCard* и *Dabo,* построены на основе таких API, как wxPython и Tkinter. При выборе соответствующей библиотеки вы также сможете использовать другие инструменты создания графического интерфейса, такие как Qt, GTK, MFC и Swing. Для разработки приложений с веб-интерфейсом или не предъявляющих высоких требований к интерфейсу можно использовать Jython (реализация Python на языке Java, описывается в главе 2) и CGI-сценарии, которые обеспечивают дополнительные возможности по созданию пользовательского интерфейса.